面向云端 FPGA 的卷积神经网络加速器的设计及其调度 *

蔡瑞初 [1]，余 洋 [1]，钟椿荣 [1]，卢 冶 [2]，陈 瑶 [1, 3†]

(1. 广东工业大学 计算机学院，广州 510006; 2. 南开大学 计算机与控制工程学院，天津 300350; 3. 新加坡高等数字科学中心，新加坡 新加坡 138602)

摘 要：卷积神经网络的高计算复杂性阻碍其广泛用于实时和低功耗应用，现有软件实现方案难以满足卷积神经网络对运算性能与功耗的要求，传统面向 FPGA 的卷积神经网络构造方式具有流程复杂、周期较长和优化空间较小等问题。针对该问题，根据卷积神经网络计算模式的特点，提出一种面向云端 FPGA 的卷积神经网络加速器的设计及其调度机制。通过借鉴基于 HLS 技术、引入循环切割参数和对卷积层循环重排的设计，采用模块化方式构造网络，并进行参数拓展以进一步优化加速器处理过程；通过分析系统任务和资源的特性总结调度方案，且从控制流和数据流两方面对其进行优化设计。与其他已有工作相比，所提出的设计提供了一种同时具有灵活性，低能耗、高能效和高性能的解决方案，并且探讨了加速器的高效通用调度方案。实验结果表明，与 CPU 实现相比，该设计实现 AlexNet 达到 8.48 倍的加速，而实现 Cifar 的功耗仅为其 24.96%；相较于 CPU+GPU 实现对 Cifar6.90 倍的加速比，虽然实现较大规模网络的性能不及 GPU，但功耗最小仅为其 14.98%；与已有研究成果相比，最大达到 6.29 倍的加速比。其中与大平台生成的加速器相比，即使仅达到相当的性能，但具有更低的时钟频率。

关键词：卷积神经网络；现场可编程门阵列；高层次综合；加速器；调度

中图分类号：TP183 doi: 10.19734/j.issn.1001-3695.2018.05.0507

Design and scheduling of convolutional neural network accelerator for cloud FPGAs

Cai Ruichu[1], Yu Yang[1], Zhong Chunrong[1], Lu Ye[2], Chen Yao[1, 3†]

(1. *College of Computer Science Guangdong University of Technology*, *Guangzhou* 510006, *China*; 2. *College of Computer & Control Engineering*, *Nankai University*, *Tianjin* 300350, *China*; 3. *Advanced Digital Sciences Center*, *Singapore* 138602, *Singapore*)

Abstract: Convolutional neural network's high computational complexity often obstructs its widespread adhibition in real-time and low-power applications. The existing software implementation solution cannot meet the demands of the convolutional neural network for computing performance and power consumption. The traditional FPGA-oriented convolutional neural network construction method has problems such as complicated process, long cycle and small optimization space. For these problems, according to the characteristics of convolutional neural network calculation pattern, this paper proposed a design and scheduling mechanism of convolutional neural network accelerator for cloud FPGAs. By using for reference the design which based HLS technology, imported the cyclic cutting parameters and rearranged the convolution layer circularly. Then constructed the network in a modular way, and extended parameters to further optimize the accelerator processing process. Summarized the scheduling scheme by analyzing the characteristics of system tasks and resources, and optimized its design from two aspects of control and data flow. In comparison with other existing work, the proposed design provided a solution with flexibility, low energy consumption, high energy efficiency and performance. The design also discussed the efficient universal scheduling scheme of the accelerator. Experimental results show that compared with the CPU implementation, this design achieves 8.84 times speedup of AlexNet, while the power consumption of Cifar implementation is only 24.96% of it. Compared with the CPU+GPU to achieve 6.90 times speedup of Cifar, although the performance of large-scale network is inferior to the GPU, but the minimum power consumption is only 14.98%. This design achieves the maximum acceleration of 6.29 times in comparison with the existing research results. Compared to the accelerators generated for large platforms, even if it only has comparable performance but with a lower clock frequency.

Key words: convolutional neural network; field programmable gate array; high-level synthesis; accelerator; scheduling

0 引言

近年来，深度学习（deep learning）的发展令人瞩目，引起了各个领域的巨大变革。以卷积神经网络（convolutional neural network，CNN）为代表的方法在图像处理领域占据统治地位[1-2]。CNN 广泛应用于计算机视觉（computer vision）[3-4]中，已经成为图像识别和分类任务中最有效的方法。有研究表明，CNN 正在取代许多传统用于视觉任务的组合算法[5]。相关的应用包

收稿日期：2018-05-22；**修回日期：**2018-08-02 **基金项目：**NSFC-广东联合基金资助项目（U1501254）；广东省杰出青年科学基金资助项目（2014A030306004）

作者简介：蔡瑞初（1983-），男，浙江温州人，教授，博士，主要研究方向为深度学习、因果关系；余洋（1993-），男，硕士，主要研究方向为神经网络、深度学习优化；钟椿荣（1993-），男，硕士，主要研究方向为神经网络、深度学习优化；卢冶（1986-），男，助理教授，博士，主要研究方向为深度学习、嵌入式系统；陈瑶（1987-），男（通信作者），博士，主要研究方向为高层次综合、深度学习优化（yaochen_nk@hotmail.com）.

括人脸识别[6]，交通标志检测[7]等。现在，CNN 已经成为众多科学领域的必备工具之一，在计算机视觉、模式识别（pattern recognition）[8]、自然语言处理（natural language processing）[9]等领域得到越来越广泛的应用。

而随着 CNN 模型朝着更大更深的方向发展，以及数据量的进一步增加，所需的存储器容量和操作数量呈指数增长。同时 CNN 在推理过程中，其计算模式具有高度重复的流水线和并行性[10]。而通用处理器为冯诺依曼结构，其串行执行指令的特性并不适合用来挖掘 CNN 的并行性，基于软件方式的 CNN 在实时性和功耗方面都不能满足应用的需求，CNN 的计算模式使其非常适合硬件加速。

基于这些原因，在基于 ASIC，GPU 和 FPGA 的文献中已经提出许多加速器[10~12]。相比于传统的 CPU 平台，GPU 虽然可以为深度学习算法提供较高的处理性能，但较高的功耗导致其计算效能低下；ASIC 解决方案虽然可达到性能和功耗之间的最佳平衡，但其高设计和制造成本使之并不具有优势；而现场可编程逻辑门阵列（field programmable gate array, FPGA）计算资源丰富、能源效率高、开发周期短、重构能力强，能够应对 CNN 运算的要求，可以充分发挥 CNN 中的并行特性[13]，并在低功耗的限制条件下，实现 CNN 的运算加速，达到性能和功耗之间的适当平衡[14]。

与嵌入式平台相比，云端 FPGA 旨在提供大量的逻辑和内存资源，提供 PCIe、DDR 控制、时钟控制等通用服务逻辑。在云环境中使用 FPGA 可实现既快速又高效节能的 CNN 推理。对于大型加速器而言，相较于嵌入式平台，云端 FPGA 中存储器和逻辑资源（DSP 除外）通常是足够的。然而，当前 CNN 算法涉及大量的权重数据，从而与输入数据计算产生大量的中间结果，FPGA 平台的片上 RAM 资源通常不足以缓冲所有数据。该限制使得应用系统需要大量的片外存储器。

然而，FPGA 设计存在较大挑战：目标硬件设计的学习难度大，开发周期长。FPGA 的容量飞速增长，使采用传统设计工具的开发效率没有以与之相对应的比率增长，所以越高的工程应用复杂性意味着越长的开发周期。高层次综合[15]（high-level synthesis, HLS）是提高 FPGA 开发效率的一种有效技术，使采用高级编程语言（例如 ANSI C/C++或者 LabVIEW）来自动生成优化后的算法执行代码对 FPGA 进行编程成为可能。

本文将介绍针对云端 FPGA 的卷积神经网络加速器的设计及其调度方案。本文的主要贡献如下：

a）构造支持 HLS 的可重用 CNN 加速器模板库函数，简化 CNN 加速器在云端 FPGA 上的设计及生成，充分发挥出 FPGA 的并行优势和云计算平台的应用优势；

b）提出通用加速器调度机制，充分利用硬件资源和片外数据传输带宽，最小化网络模型处理延迟。

1　相关工作

CNN 作为乘法累加(multiply-accumulate, MAC)密集型算法，其固有的并行性使其实现方式与并行计算紧密相关。神经网络应用对于性能和功耗的需求，使 CNN 如何在低功耗条件下发挥高能效的问题得到学者广泛关注和深入研究。GPU 已经被证实可以达到足够的性能水平[2,11,16]；然而其高功耗和低能效使之不能满足应用的加速需求。而 ASIC 解决方案[17]需要高设计和制造成本，且灵活性较低。在这些用于硬件加速的不同类型的器件中，FPGA 成为硬件加速的理想选择。

文献[13]中使用 Zynq SoC 实现了一种采用 8 路并行引擎的 CNN 加速器。但该加速器面向卷积核设计，当卷积核尺寸变化时，不能充分发挥并行计算性能，会造成大量的资源浪费；Antony 等人[18]通过 FPGA 实现了多层感知器网络，并详细分析了定点数格式表示和浮点数格式表示以及串行实现和并行实现对硬件实现的神经网络的性能的影响。然而，该实现并未给出具体完整的 CNN 的 FPGA 设计方案。

通过使用循环优化技术[19]和良好的缓存设计[20]，CNN 可以表现出确定性的数据访问模式，因此可以最大化数据重用并减少片外数据传输。在[12]的工作中，Zhang 等人使用多面体方法评估这些采用循环嵌套优化的技术，使用 roofline 模型[21]解决寻找上述优化的有效平衡的问题，以提高性能并减少 FPGA 面积。但是仅用其方法测试网络[2]的第 1-5 层，通过避免评估通信密集型的全连接层（第 6-8 层）而人为产生高 CTC 比率（计算通信比）。文献[22]中也展示了仅 1-5 层的类似测试，这意味着全连接层将由主机处理，而不是采用卷积层和全连接层通用的加速器设计。

Murugan 等人[23]设计了使用 FPGA 中大量 DPS 单元和片外 DDR 存储器的 CNN 协处理器，为了实现复杂的运算模块控制，该设计仍然需要使用 PC 作为上位机。同时，上位机也参与部分运算。与上述前几个相关工作一样，对于系统复杂的逻辑和模块控制，并未考虑设计良好的调度方案。

2　HLS 及卷积神经网络

2.1　HLS 技术的优势和限制

传统地采用寄存器传输级（register transfer level，RTL）描述语言设计 FPGA 代码具有流程复杂、周期较长和优化空间较小等问题。借助 HLS 技术，可以从硬件和软件两个角度快速探索设计空间，给 FPGA 设计带来很大便利，在不影响性能的情况下大幅缩短 FPGA 研发周期，使其开发时间甚至低于 DSP 和 GPU[24]。基于 HLS 的优化是通过使用诸如指定流水线，展开循环和处理数据传输之类的技术来提高总体设计效率的实用方式。HLS 非常高效，允许使用 C，C++和 OpenCL 等高级语言来设计 FPGA 硬件加速器。与 C++等高级编程语言相比，HLS 要求在编译之前将内存需求和目标函数的接口声明为静态。某些 C/C++语法不能被综合，如一些依赖操作系统的函数、动态内存分配和标准模板库等。为了使设计高效可用，本文充分考虑了这些限制条件。

2.2　CNN 模型功能层

典型的卷积神经网络由卷积层、池化层、全连接层以及激活层构成。如图 1 所示。

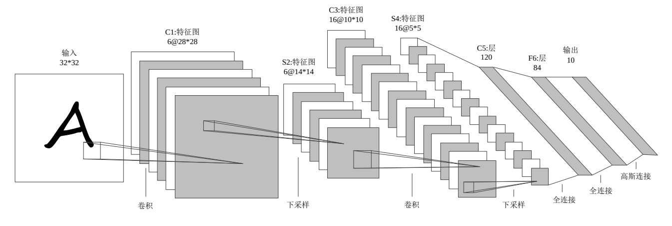


图 1　典型卷积神经网络模型

Fig.1　Typical convolution neural network model

卷积层输入特征图通过和卷积核局部连接进行卷积操作，将卷积结果累加并加一个偏置量后所得即为输出特征图。通过该过程从输入图像提取不同特征，并且通过堆叠多个卷积层来提取更高级特征。一般地，卷积层可以用以下表达式表示：

$$X_j^l = \sum_{i \in N_j} X_i^{l-1} * W_{i,j} + B_j^l \tag{1}$$

其中: X_j^l 表示第 l 层卷积层的第 j 个卷积核对应的输出特征图，N_j 表示当前卷积对输入特征图的选择，$W_{i,j}$ 表示第 l 层的第 j 个卷积核的第 i 个加权系数，B_j^l 表示第 l 层卷积层的第 j 个卷积核对应的偏置系数。

池化层的工作原理类似于卷积层。池化层与卷积层相连，可起到二次特征提取的作用。通常采用最大值或均值采样进行池化操作来降低输入矩阵的规模。池化操作可有效减小特征映射的尺寸以节省大量的内存来存储中间结果；且其像素选择可确保将最相关的特征传播到下一层。

全连接层也称为线性层，是一种特殊的卷积层。全连接层可整合卷积层或池化层中区分类别的局部信息，通过对输入进行线性空间转换，从而得到向量形式输出，网络最后一个线性层的输出特征图数等于要识别的类别数。

激活层中激活函数的应用将像素值压缩至特定函数范围内，避免因卷积层中的乘法累加操作而导致值无限增加。且该函数通过添加非线性，将分类边界平滑化。常用的激活函数包括冲击响应（Sigmoid）、非线性（ReLU）、三角函数（Tanh）等。卷积层经过激活函数的非线性变换可表示为式(2)，f 为激活函数。

$$X_j^l = f(\sum_{i \in N_j} X_i^{l-1} * W_{i,j} + B_j^l) \tag{2}$$

3　基于 HLS 的模板函数库设计

本章主要针对 CNN 网络模型，对文献[12]中的设计方法进一步性能优化，通过模块化方式构造网络，提高计算的并行度和资源利用效率，增强加速器模板的通用性。

3.1　性能优化

典型的卷积层由 6 级 for-loops 组成[12]。本文借鉴[12]中的设计，即采取分块实现方式，为卷积层加速器引入循环切割参数，分别对输入输出维度的循环进行切割，通过对卷积层循环的重排，使得其可以借助 HLS 工具所提供的循环展开及流水线优化，进行硬件生成优化。最后采用缓存优化指令，使得卷积计算单元的执行与缓存单元获取数据并行执行，从而提高加速器内部的并行度。在此基础上，在卷积层加速器进行高层次综合前，将其数值确定为固化的加速器参数，并进行参数拓展，根据 FPGA 生成的加速器硬件地址映射特点设计 offset 参数来表示地址偏移量。使用地址偏移量可以提高 FPGA 片上资源利用效率，增加数据缓存大小、减少数据缓存次数，从而加快卷积计算的进程。优化后的卷积计算过程如图 2 所示。

```
in_buf [Tn][I_BUF][I_BUF];
w_buf [Tn][Tm][W_BUF][W_BUF];
out_buf [Tm][O_BUF][O_BUF];
load weight();
load bias();
load input();
for (i = 0; i < K; i++) {
  for (j = 0; j < K; j++) {
    for (tr = 0; tr < Tr; tr++) {
      for (tc = 0; tc < Tc; tc++) {
#pragma HLS PIPELINE
        for (tm = 0; tm < Tm; tm++) {
#pragma HLS UNROLL
          for (tn = 0; tn < Tn; tn++) {
#pragma HLS UNROLL
            out_buf[tm][tr + r_offset_o][tc + c_offset_o] =
              w_buf[tn][tm][i + w_r_offset][j + w_c_offset] *
              in_buf[tn][S * tr + i + r_offset_i][S * tc + j + c_offset_i];
} } } } } }
```

图 2　优化后的卷积计算

Fig.2　Optimized convolution calculation

由于卷积计算与池化计算均需大量的循环，且其数据获取及计算过程具有相似性，以上卷积加速器的优化方式同样适用于池化层加速器，但由于池化不改变输入特征维度，所以池化层加速器共用输入输出特征维度。

3.2　模板化构造网络层功能

由于 CNN 前向传播过程中，层间运算具有独立性且相同类型的层运算模式相同，因此可通过设计针对不同类型层的通用的模板函数来实现 CNN 前向计算。而 CNN 前向传播是按照网络结构逐层顺序进行，即网络相邻层之间有数据依赖，不能并行处理。据此，基于上文的性能优化设计，通过设计足够通用的网络层模板，可以复用单层运算资源来节约硬件资源，让加速器以更高的计算能效来实现完整的 CNN 前向计算。

为了让不同规模的 CNN 均可复用硬件运算单元，需对硬件运算单元进行灵活性设计，并将配置参数化，使加速器模板具有可依据不同网络模型进行配置重构的能力。其中，加速器所采用数据类型、数据位宽、加速器缓存大小、数据处理并行度等，在编程实现时，均为加速器模板参数；同时数据获取方式、数据处理模块、数据输出模块等也进行参数化，可依据应用进行调整与重构。

a）卷积层。卷积层经过输入特征图和卷积核权重的三维乘法和累加得到输出，所处理的输入输出数据均为 3 维数据。基于卷积层的计算性质和分块实现方式，设计参数列表为输出、输入特征数，输出特征大小，内核大小，输入、输出特征偏移量，步长大小，填充大小和激活类型。根据 FPGA 生成的加速

器硬件地址映射特点，设计 offset 参数来表示偏移量。卷积加速器模板如图 3 所示。

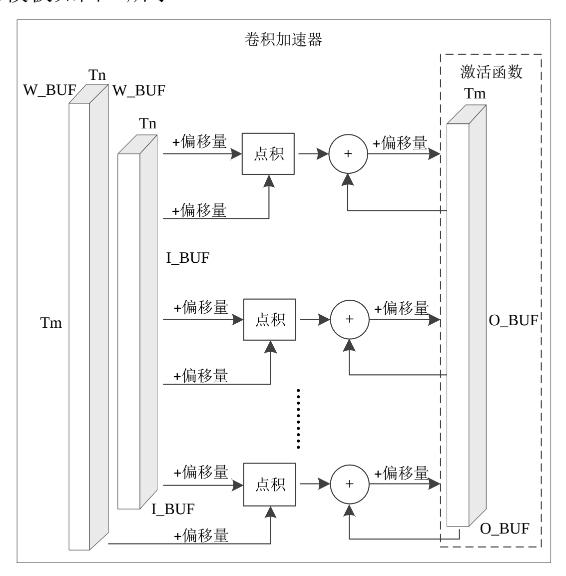


图 3　卷积加速器模板

Fig.3　Convolution accelerator template

其中，Tn、Tm 为输入、输出缓冲区特征数，I_BUF、O_BUF 为输入、输出缓冲区大小，W_BUF 为权重缓冲区大小。

b）池化层。池化层采取与卷积层类似的滑动窗口处理输入数据，并对每个滤波器的输入求最大值或平均值等。考虑到不同网络中卷积层和池化层的连接关系不同，将池化层以与卷积加速器相同的方式实例化为独立加速器，来增强加速器对不同模型的适用性与可重构性。采取与卷积层相同的加速器设计，由于池化层不改变输入特征维度，因此只有一个特征维度参数，其参数定义与卷积加速器相同，如图 4 所示。

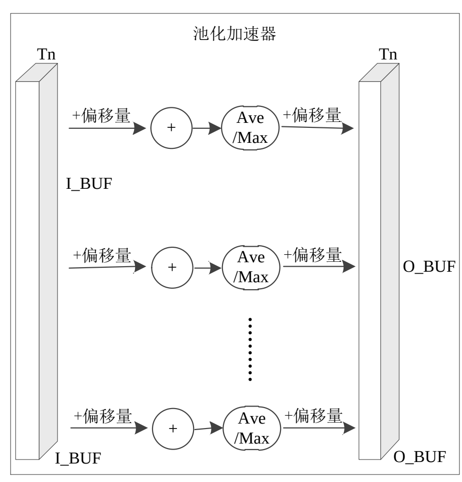


图 4　池化加速器模板

Fig.4　Pooling accelerator template

c）全连接层。通过分析 CNN 的特性得知，全连接层的计算相对于卷积层，区别仅在于减少了卷积核心内部的循环操作，可将全连接层视为卷积层的一种特殊形式。因此，采用卷积层加速器进行全连接层的计算，可通过复用卷积加速器来节约硬件资源，提高加速器的计算效能，达到对资源的高效利用。

d）数据存储体。为上述加速器分配相应的数据存储体，同样将其参数化。数据存储体用于存储 3 维输入和输出数据，以及相应的权重和偏置。采用多存储接口，设计输入、输出存储体的参数列表为输入、输出缓冲区特征数，输入、输出缓冲区大小；权重存储体的参数列表为输入、输出缓冲区特征数，权重缓冲区大小，其结构如图 3 中的 3 维数据存储体。

4　加速器的调度机制

本章主要介绍基于前文加速器模板设计的 CNN 加速器系统架构和相应通用加速器调度机制，即软硬件划分、系统构造流程、任务-资源调度模型和相应的调度优化方案。

4.1　系统架构

4.1.1 软-硬件划分

本文采用软硬件协同设计。结合 HLS 技术与云端 FPGA 平台所实现的“处理器+加速器”架构，通过对任务在软硬件上的划分与协同设计，完成任务模型向硬件平台的映射。云端 Host-CPU 上的软件部分作为整个系统的控制端。CPU 运行软件代码，并为硬件分配加速器任务。硬件端可根据不同加速任务，启动加速器来实现对计算的加速。CNN 应用的执行过程主要分为三个阶段，即数据预处理、CNN 推理、分类结果的整理和输出。其中，CNN 推理中卷积、池化、全连接和激活函数等操作计算需求高，且数据吞吐率相对较高，将其划分为硬件加速器实现；而另外两个阶段与应用的前后端联系更为紧密，计算需求低，接口要求高，将其划分为 Host-CPU 执行。此外，加速器状态的控制、数据的转变和传输、CNN 应用执行时间的测量和系统的调试也由 Host-CPU 管理。目标系统的软硬件划分如图 5 所示。

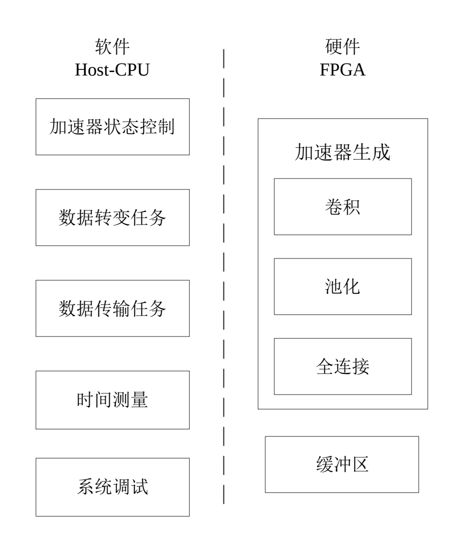


图 5　系统软硬件划分

Fig.5　System hardware and software partition

4.1.2 系统构造流程

构建 CNN 应用首先需要掌握网络模型的配置和权重信息，通过构造 CNN 模型分析器，对使用 Caffe 深度学习框架[25]训练所得网络模型描述文件进行分析，将其提取为网络配置参数和权重数据文件。其中，网络配置参数文件包括网络结构，所有层类型和输入、输出特征尺寸，卷积核大小，步长大小，填充

大小等信息。然后根据系统软硬件划分，完成任务模型向软硬件端的映射。通过提取网络模型特征，结合 CNN 加速器模板，实现加速器的设计与生成。针对 FPGA 片上资源结构及特征，实现加速器缓冲区的构建及生成。最后，完成 CPU 和 FPGA 之间的任务分派及其任务-资源调度。系统构造流程如图 6 所示。

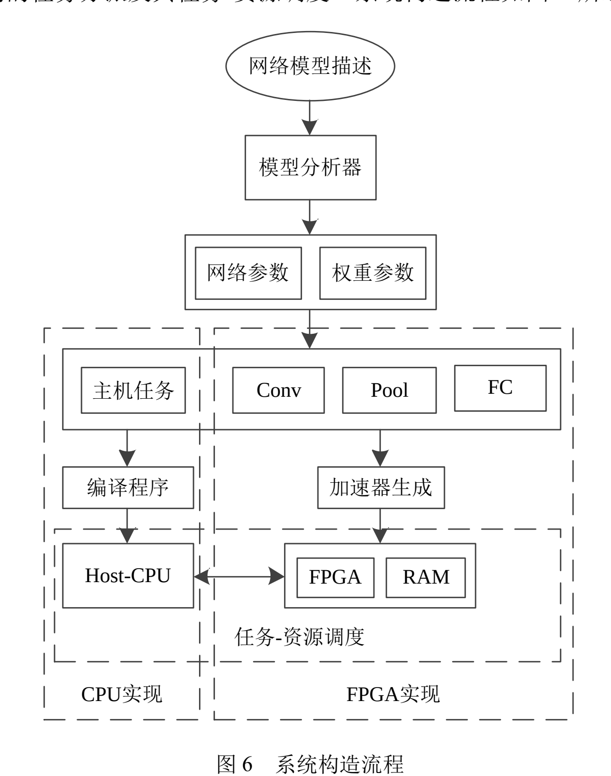


图 6　系统构造流程

Fig.6　System construction process

4.2　任务-资源调度模型

4.2.1 任务调度

基于前文的软-硬件协同设计和 CNN 数据访问模式的特性建模分析得知，在 FPGA 上实现 CNN 各层的处理过程可分为三个操作阶段，即输入、计算和输出阶段。

a）输入阶段。将计算所需的输入数据从 CPU 片外 DRAM 移至 FPGA 片上 BRAM。网络权重数据已在预处理阶段传输完成，不计入该输入阶段。通过将输入数据的一部分突发读入临时缓冲区，并移动到加速器生成的输入缓冲区中。

b）计算阶段。运算单元在启动时，从宿主端读取配置信息，根据配置信息，调用加速器启动命令执行计算。该阶段紧接于输入阶段处理所需的所有后续计算。在输入和权重之间执行元素乘法和累加计算，输出结果存储于加速器输出缓冲区。

c）输出阶段。将上阶段加速器输出缓冲区所得结果传回 CPU 片外存储器，并将其作为网络下层的输入数据。

由于采用分块实现方式，系统的逻辑和模块控制比较复杂，所以需对 CNN 各层的任务进行调度。系统中的 Host-CPU 负责加速器状态与运算过程的控制以及数据的传输，由主机控制代码完成。整个推理过程被封装为一个任务单元并分配给网络加速器。网络加速器被封装为 PCIe 设备，应用程序调用集不需要知道硬件的细节。根据以上分析，图 7 展示了执行卷积层的任务调度过程。

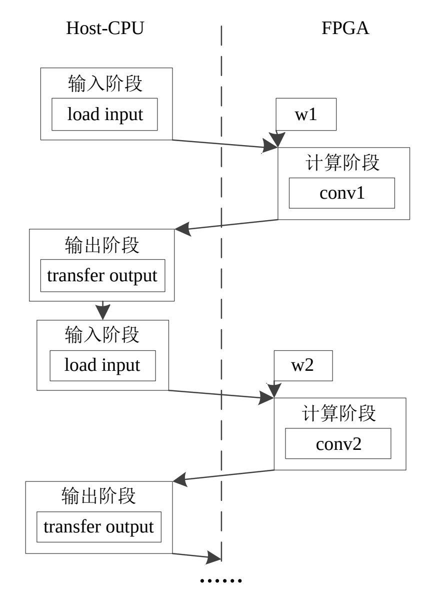


图 7　任务调度

Fig.7　Task scheduling;

4.2.2 资源调度

虽然将模型参数和中间结果存储在片上 BRAM 中可以显著提高性能，但 PFGA 由于其计算资源、传输带宽的限制，往往无法满足高并行性所需的资源要求。因此，必须在片外动态存储器中分配数据缓冲区来存储中间结果和模型参数。将网络整体的资源划分为计算资源和缓存资源。加速器分配的层计算参数、权重、输入和输出缓冲区属于计算资源，缓存资源则用来存储模型参数和中间结果。资源调度的整体控制由通用处理器完成，当任务调度的输入阶段开始执行，从 CPU 片外存储器传输数据至加速器输入缓冲区，然后控制启动加速器进行计算。计算完成后，将加速器输出缓冲区的结果传输至 CPU 片外存储器。对应于任务调度三阶段中的输入和输出阶段，两种资源之间数据传输的方向分别映射为缓存资源→计算资源，计算资源→缓存资源。根据以上描述，总结资源调度过程如图 8 所示。

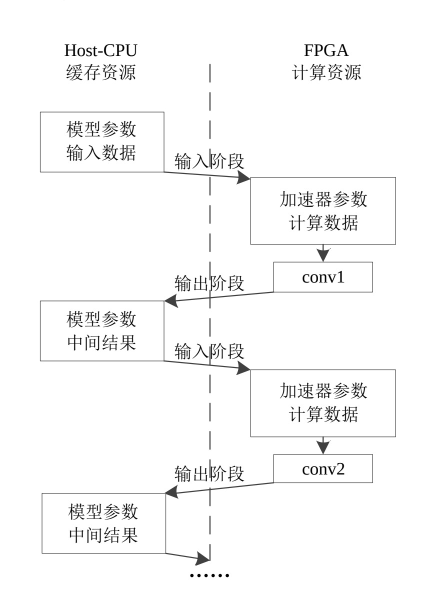


图 8　资源调度

Fig.8　Resource scheduling;

4.3 调度优化

4.3.1 数据流

在卷积过程中，输入特征图上相同滤波器中不同位置之间的部分像素相互重叠，使得有必要在不同的迭代中存储一些输入像素。由此考虑在硬件资源和带宽允许的情况下，在每个卷积层的输入阶段向加速器输入缓冲区传输部分数据。而该阶段传输数据消耗的时间计入网络处理时间，需对其进行优化来减小网络处理的延迟。通过每次仅存储计算输出特征图的 Tr 行所需的一部分输入数据可以最小化输入阶段等待时间。设计为存储输入的行数为 Tr * S，并且列的数量为 Tc * Tn，其中 Tr 和 Tc 是加速器设置的输出特征大小，S 和 Tn 分别是该层的步长大小和输入特征缓冲区的通道数量。图 9 说明了优化后的操作过程。在开始时，窗口的第一个 Tr * S 行被填充以便有足够的数据来开始计算。当卷积核在该 Tr * S 行上滑动完毕，新的一个 Tr * S 行输入数据被加载到窗口中进行下一次迭代，并以相同方式继续计算，直到产生所有输出特征映射。

片上缓冲区的设计基于 Double Buffering 的基本思想，来保证数据并行读取计算需求。其中双缓冲区采用 ping-pong 操作，以便将数据传输时间与计算重叠，从而提高处理效率。将片上缓冲区分为两组，分别用于输入特征图和输出特征图。每个缓冲区集都包含多个独立的地址连续的缓冲区，每个输入缓冲区集和输出缓冲区集中的缓冲区组数量分别等于 Tn 和 Tm。

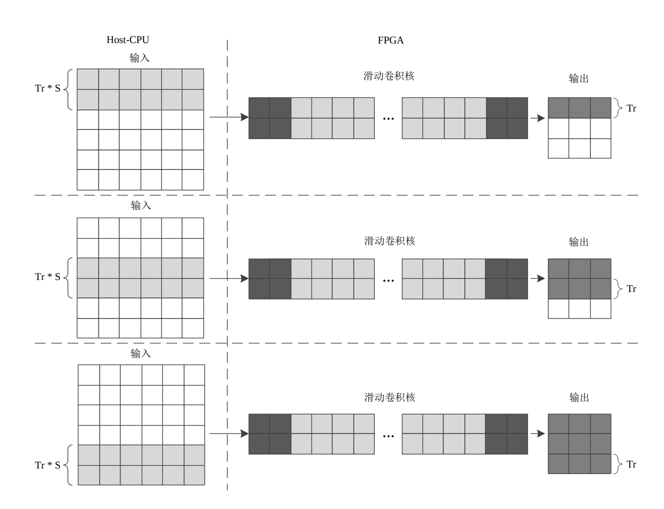


图 9 卷积数据流调度优化

Fig.9 Scheduling optimization of convolution data stream scheduling optimization

图 10 举例显示了几个计算和数据传输阶段的时序。对于输入特征映射，第一阶段，计算引擎正在处理 input0，同时将下一个输入阶段所需的数据复制到 input1,下一个输入阶段将执行相反的操作。对于输出特征映射，当计算和数据复制的⌈N/Tn⌉个阶段完成后，得到的输出特征映射存储于输出缓冲区集中。当 output0 的空间被最大化利用后，则使用 output1，并传输 output0 中存储的输出数据，下一个输出阶段同样执行相反的操作。

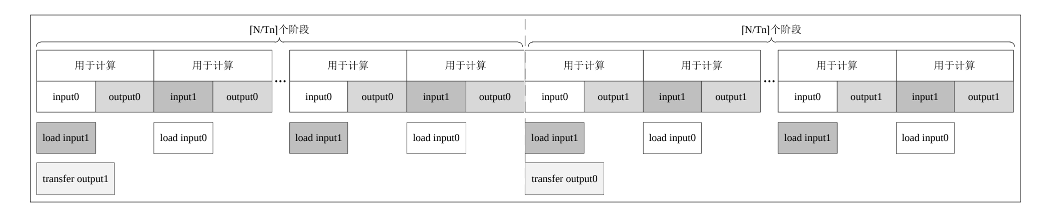


图 10 基于 Double Buffering 的数据流调度优化

Fig.10 Dataflow scheduling optimization based on Double Buffering

4.3.2 控制流

在云端 FPGA 平台实现 CNN 推理的过程，包含软件和硬件实现两部分任务。在三个任务阶段中，输入和输出阶段由软件端控制实现，计算阶段则由软件端调用加速器启动命令来执行计算，即启动后的计算过程由硬件端控制实现。而软件和硬件执行部分的操作和资源都互相独立，且本文采用 Double Buffering 设计，这使得当前计算阶段与下个输入阶段及与上个输出阶段均可达到一定程度的并行化，从而减小整体网络延迟。对应于数据流优化，相应的控制流进行调度优化后的任务调度流程如图 11 所示。

5 评估

5.1 实验设置

5.1.1 软硬件环境

本文实验采用配备一个 Xilinx VU9P FPGA 板卡的 AWS F1 EC2 实例作为基于云的实验平台。该 FPGA 包含 2585K 逻辑单元和 6800 个 DSP。此外，它拥有 75.9Mb 的 Block RAM 和 270 Mb UltraRAM。网络模型的软件实现运行在 8 核英特尔 i7-Xeon CPU E5-2430（带 15MB 缓存）和 NVIDIA GPU Tesla K80 上。

5.1.2 网络模型与测试数据集

选取 Cifar、AlexNet、VGG-16 等 CNN 模型及其量化版本作为实验对象，量化方法基于文献[26]，在本文的评估中，量化版本的输入数据和权重都被量化为 16 bit。将 CiFar-10 数据集和 2012 年 ImageNet 大规模视觉识别挑战赛（ImageNet Large Scale Visual Recognition Challenge，ILSVRC）的验证集作为实验的测试数据集。

5.2 性能评估

实验中对以下几方面进行评估。

a）评估属性。对设计生成的加速器进行评估，分析几种网络的处理性能、能耗和能效等。

b）评估对象。展示生成的系统性能，并将其分别与软件实现和其他已有设计的各项性能进行比较分析。

5.3 结果与分析

a）首先评估针对这几个网络模型设计生成的加速器性能。

通过测量在云端 FPGA 平台实现中运行网络推理的处理时间来衡量整体性能。通过分析网络参数，表 1 中测量并报告了各网络模型的加速器参数设置和性能。

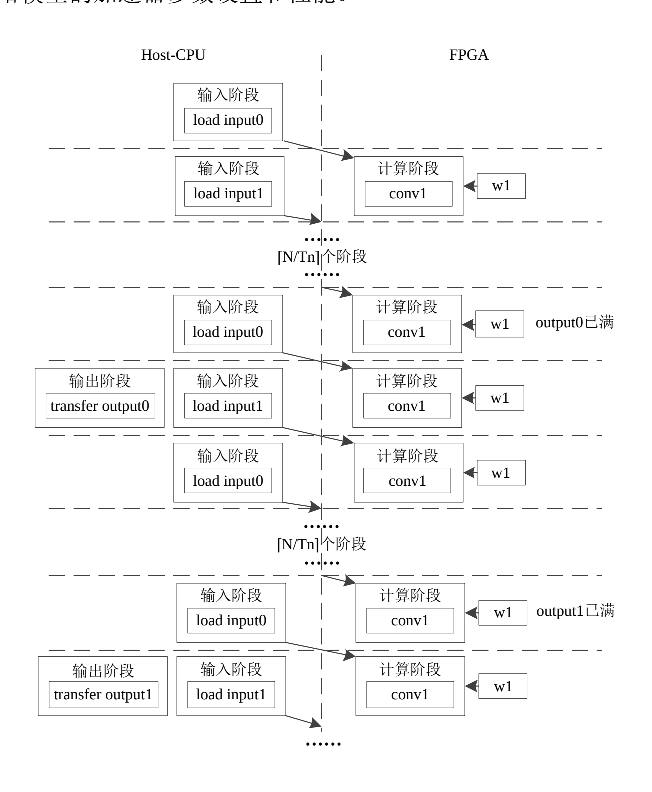


图 11　控制流调度优化

Fig.11　Scheduling optimization of Control flow

表 1　各网络模型的加速器配置和性能

Table 1　Accelerator configuration and performance of network models

网络模型	数据类型	加速器配置（Tm, Tn, Tr, Tc, I_BUF）	时间（ms/image）
Cifar	float	32,32,4,4,5	0.42
AlexNet	float	133,10,4,4,19	28.6
VGG-16	float	64,19,4,4,32	224.85

b)将本文实现的网络模型的性能与其纯软件版本进行比较。结果如表 2 所示。所有网络模型的加速都将 CPU 实现测得的延迟数据作为 GPU 和 FPGA 系统的基线来展示加速比。GPU 和 FPGA 的能耗和能效也与 CPU 进行比较。

由表 2 可见，对于小型网络，CPU 可以实现与 GPU 相当的性能。但是，利用架构的灵活性，FPGA 可以优于 CPU 和 GPU。与 CPU 实现相比，该设计实现 AlexNet 达到 8.48 倍的加速，而实现 Cifar 的功耗仅为其 24.96%；相较于 CPU+GPU 实现对 Cifar 6.90 倍的加速，虽然实现较大规模网络的性能不及 GPU，但功耗最小仅为其 14.98%。据此分析得知，相比于 CPU 实现，得益于 FPGA 较高的并行执行能力，生成的系统在性能和能效方面总是更佳；与 GPU 实现相比，即使本文设计生成的加速器系统性能达不到 GPU 的性能峰值，也仍然能利用 FPGA 的高计算密度优势提高能效，表现出较高的能耗优势。

c)将本文实现的网络模型的性能与已发表的研究成果进行比较。结果展示在表 3 中。

表 2　与其纯软件版本比较结果

Table 2　Compare results with its software version

平台	CPU			CPU+GPU			本文（CPU+FPGA）		
设备	E5-2430			E5-2609 + K80			AWS F1		
网络模型	Cifar	AlexNet	VGG-16	Cifar	AlexNet	VGG-16	Cifar	AlexNet	VGG-16
数据类型	float			float16			float		
时钟（MHz）	1.9GHz			1GHz			125MHz		
能耗（Watt）	150			250			37.447	42.41	40.432
延迟/图（ms）	2.4065	242.562	794.238	2.8979	5.0486	25.7583	0.42	28.6	224.85
加速比（×）	1	1	1	0.83	48.05	30.83	5.73	8.48	3.61
能效	1	1	1	0.4034	28.203	16.95	17.292	28.6	12.11

表 3　与已有工作比较结果

Table 3　Compare results with the existed methods

设计来源	文献[27]	文献[28]	文献[28]	文献[29]	本文
平台	VX690T	Arria 10	Stratix V	Stratix V	VU118
网络模型	VGG-16	VGG-16	VGG-16	VGG-16	VGG-16
数据类型	fixed16	fixed16	fixed16	fixed8-16	fixed16
时钟（MHz）	150	200	150	120	125
DSP 利用率（%）	78.7	100	100	37	88.47
能耗（Watt）	26	-	-	-	42.241
延迟/图（ms）	65.13	65.13	87.87	262.9	59.9（41.77）



本文设计生成的加速器系统，在每张图像处理延迟和能源效率方面与其他已有工作相当甚至更有优势。相较于已有工作，最大达到 6.29 倍的加速比。与在较小平台上设计的加速器相比[27]，本文设计生成的加速器系统充分利用片上资源实现了更好的性能。其中与大平台生成的加速器相比[28-29]，本文设计生成的系统具有相当或更佳的性能，即使仅达到相当的性能，但具有更低的时钟频率。

6　结束语

本文根据卷积神经网络计算模式高度重复的流水线和并行性的特点，提出一种面向云端 FPGA 的卷积神经网络加速器的设计及其调度机制。实验结果表明，该加速器可在有效提高运算速度的同时减小功耗。对 Cifar、AlexNet、VGG-16 等网络模型采用本文设计系统实现的评估结果表现出与已有工作相当或更佳的性能，并且与 CPU 和 GPU 相比，在云端 FPGA 上的功耗及性能均表现出较大优势。

未来工作中，还将探索多种加速器之间拆分推理的设计可能性。本文已有的工作采用模块化设计，系统可拓展性较高，各种新型计算类型可轻松集成到设计中。未来将进一步扩展笔者的设计以支持新功能，并为 CNN 的 FPGA 映射提供更多的自适应解决方案。

参考文献：

[1] 周飞燕，金林鹏，董军. 卷积神经网络研究综述 [J]. 计算机学报, 2017, 40 (6): 1229-1251. (Zhou Feiyan, Jin Linpeng, Dong Jun. Review of convolutional neural network [J]. Chinese Journal of Computers, 2017, 40 (6): 1229-1251)

[2] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks [C]// Advances in Neural Information Processing Systems. New York: ACM Press, 2012: 1097-1105.

[3] Saxena S, Verbeek J. Heterogeneous face recognition with CNNs [C]// Proc of the European Conference on Computer Vision. Berlin: Springer Press, 2016: 483-491.

[4] Ji Shuiwang, Xu Wei, Yang Ming, *et al.* 3D convolutional neural networks for automatic human action recognition: USA, US8345984 [P/OL]. (2010-06-11) [2013-01-01]. http://www. google. com/patents/US8345984.

[5] Shao Hong, Chen Shuang, Zhao Jieyi, *et al.* Face recognition based on subset selection via metric learning on manifold [J]. Frontiers of Information Technology & Electronic Engineering, 2015, 16 (12): 1046-1058.

[6] Levi G, Hassncer T. Age and gender classification using convolutional neural networks [C]// Proc of Computer Vision and Pattern Recognition Workshops. Piscataway, NJ: IEEE Press, 2015: 34-42.

[7] Peemen M, Mesman B, and Corporaal H. Speed sign detection and recognition by convolutional neural networks [C]// Proc of the 8th International Automotive Congress. 2011: 162–170.

[8] Lee W Y, Park S M, Jang I H, *et al.* CNN-based shoe-upper pattern recognition and generation of adhesive point [J]. Journal of Institute of Control Robotics & Systems, 2017, 23 (9): 725-731.

[9] 蔡慧苹，王丽丹，段书凯. 基于 word embedding 和 CNN 的情感分类模型 [J]. 计算机应用研究, 2016, 33 (10): 2902-2905. (Cai Huiping, Wang Lidan, Duan Shukai. Sentiment classification model based on word embedding and CNN [J]. Application Research of Computers, 2016, 33 (10): 2902-2905.)

[10] Sankaradas M, Jakkula V, Cadambi S, *et al.* A massively parallel coprocessor for convolutional neural networks [C]// Proc of IEEE International Conference on Application-Specific Systems, Architectures and Processors. Piscataway, NJ: IEEE Press, 2009: 53-60.

[11] Potluri S, Fasih A, Vutukuru L K, *et al.* CNN based high performance computing for real time image processing on GPU [C]// Proc of Workshop on Nonlinear Dynamics & Synchronization & Intl Symposium on Theoretical Electrical Engineering. Berlin: Springer Press, 2011: 1-7.

[12] Zhang Chen, Li Peng, Sun Guangyu, *et al.* Optimizing FPGA-based accelerator design for deep convolutional neural networks [C]// Proc of ACM//SIGDA International Symposium on Field-Programmable Gate Arrays. New York: ACM Press, 2015: 161-170.

[13] Cadambi S, Majumdar A, Becchi M, *et al.* A programmable parallel accelerator for learning and classification [C]// Proc of International Conference on Parallel Architectures and Compilation Techniques. New York: ACM Press, 2010: 273-284.

[14] Wei Xuechao, Yu Codyhao, Zhang Peng, *et al.* Automated systolic array architecture synthesis for high throughput CNN inference on FPGAs [C]// Proc of Design Automation Conference. ACM, New York: ACM Press, 2017: 29.

[15] Farrens M, Chong F T, Oskin M. HLS: combining statistical and symbolic simulation to guide microprocessor designs [C]// Proc of International Symposium on Computer Architecture. New York: ACM Press, 2000: 71-82.

[16] Strigl D, Kofler K, Podlipnig S. Performance and scalability of GPU-based convolutional neural networks [C]// Proc of Euromicro International Conference on Parallel, Distributed and Network-Based Processing. Piscataway, NJ: IEEE Press, 2010: 317-324.

[17] Ferry C. Hardware accelerated convolutional neural networks for synthetic vision systems [C]// Proc of IEEE International Symposium on Circuits and Systems. Piscataway, NJ: IEEE Press, 2010: 257-260.

[18] Savich A W, Moussa M, Areibi S. The impact of arithmetic representation on implementing MLP-BP on FPGAs: A study [J]. IEEE Trans on Neural Networks, 2007, 18 (1): 240-252.

[19] Peemen M M, Mesman B B, Corporaal H H. Optimal iteration scheduling for intra-and inter-tile reuse in nested loop accelerators [D]. Eindhoven: Eindhoven University of Technology, 2013.

[20] Peemen M, Setio A A A, Mesman B, *et al.* Memory-centric accelerator design for convolutional neural networks [C]// Proc of IEEE, International Conference on Computer Design. Piscataway, NJ: IEEE Press, 2013: 13-19.



[21] Williams S, Waterman A, Patterson D. Roofline: An insightful visual performance model for floating-point programs and multicore architectures [J]. Office of Scientific & Technical Information Technical Reports, 2009, 52 (4): 65-76.

[22] Motamedi M, Gysel P, Akella V, *et al.* Design space exploration of FPGA-based deep convolutional neural networks [C]// Proc of Design Automation Conference. Piscataway, NJ: IEEE Press, 2016: 575-580.

[23] Gokhale V, Jin Jonghoon, Dundar A, *et al.* A 240 G-ops/s mobile coprocessor for deep neural networks [C]// Proc of IEEE Conference on Computer Vision and Pattern Recognition Workshops. Piscataway, NJ: IEEE Press, 2014: 696-701.

[24] Baklouti M, Ammar M, Marquet P, *et al.* A model-driven based framework for rapid parallel SoC FPGA prototyping. [J]. Journal of Biological Chemistry, 2011, 272 (12): 7797-800.

[25] Jia Yangqing, Shelhamer E, Donahue J, *et al.* Caffe: Convolutional architecture for fast feature embedding [C]// Proc of ACM International Conference on Multimedia. New York: ACM Press, 2014: 675-678.

[26] Gysel P, Motamedi M, Ghiasi S. Hardware-oriented approximation of convolutional neural networks [J]. arXiv preprint arXiv: 1604. 03168, 2016.

[27] Zhang Chen, Fang Zhenman, Zhou Peipei, *et al.* Caffeine: towards uniformed representation and acceleration for deep convolutional neural networks [C]// Proc of International Conference on Computer-Aided Design. New York: ACM Press, 2016: 12.

[28] Ma Yufei, Cao Yu, Vrudhula S, *et al.* An automatic RTL compiler for high-throughput FPGA implementation of diverse deep convolutional neural networks [C]// Proc of International Conference on Field Programmable Logic and Applications. Piscataway, NJ: IEEE Press, 2017: 1-8.

[29] Qiu Jiantao, Wang Jie, Yao Song, *et al.* Going deeper with embedded FPGA platform for convolutional neural network [C]// Proc of ACM//SIGDA International Symposium on Field-Programmable Gate Arrays. New York: ACM Press, 2016: 26-35